Nonlinear Shape Prior from Kernel Space for Geometric Active Contours

Samuel Dambreville, Yogesh Rathi and Allen Tannenbaum

Georgia Institute of Technology, 30332, Atlanta, Georgia, USA

ABSTRACT

The Geometric Active Contour (GAC) framework, which utilizes image information, has proven to be quite valuable for performing segmentation. However, the use of image information alone often leads to poor segmentation results in the presence of noise, clutter or occlusion. The introduction of shapes priors in the contour evolution proved to be an effective way to circumvent this issue. Recently, an algorithm was proposed, in which linear PCA (principal component analysis) was performed on training sets of data and the shape statistics thus obtained were used in the segmentation process. This approach was shown to convincingly capture small variations in the shape of an object.

However, linear PCA assumes that the distribution underlying the variation in shapes is Gaussian. This assumption can be over-simplifying when shapes undergo complex variations. In the present work, we derive the steps for using Kernel PCA to in the GAC framework to introduce prior shape knowledge. Several experiments were performed using different training-sets of shapes. Starting with any initial contour, we show that the contour evolves to adopt a shape that is faithful to the elements of the training set. The proposed shape prior method leads to better performances than the one involving linear PCA.

Keywords: Shape Prior, Active Contours, Principal component Analysis, Kernel Methods

1. INTRODUCTION

Segmentation consists of extracting an object from an image, an ubiquitous task in computer vision applications. It is quite useful in applications ranging from finding special features in medical images to tracking deformable objects.[1–5] The active contour methodology has proven to be quite valuable for performing this task. However, the use of image information alone often leads to poor segmentation results in the presence of noise, clutter or occlusion. The introduction of shape priors in the contour evolution process has been shown to be an effective way to address this issue, leading to more robust segmentation performances.

Many different methods which use a parameterized or an explicit representation for contours have been proposed.[6–8] Cremers *et al.*[9] used B-spline parametrization to build shape models in the kernel space.[10] These models were then used in the segmentation process to provide shape prior. The geometric active contour framework[11] (GAC) involves a parameter free representation of contours, i.e., a contour is represented implicitly by the zero level set of a higher dimensional function, typically a signed distance function.[12] The advantage of implicit representations (e.g.: level sets) over explicit representations (e.g.: B-spline) is notably that changes in the topology of the contour are handled in a natural fashion: In particular, the contour can split and merge during evolution without any particular treatment. To introduce shape prior within the GAC framework, Leventon *et al.*[2] obtained shape statistics using linear principal component analysis (PCA) on training sets of signed distance functions (SDFs). This approach was shown to be able to convincingly capture small variations in the shape of a unique object. This method, involving linear PCA, inspired other schemes to obtain shape prior.[3, 13]

However, linear PCA assumes that the distribution underlying the variation in shapes is Gaussian. This assumption can be over-simplifying when shapes undergo complex variations. Hence, linear PCA can lead to shapes that are unrealistic elements of the sub-manifold induced by the learnt shapes. Cremers *et al.*, successfully pioneered the use of kernel methods to address this issue, within the GAC framework.[14] In the present work, we propose to use Kernel PCA to introduce shape

Further author information: (Send correspondence to S.D.)
S.D.: E-mail: samuel.dambreville@bme.gatech.edu, Telephone: 1 404 385 5062

Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning
edited by E.R. Dougherty, J.T. Astola, K.O. Egiazarian, N.M. Nasrabadi, S.A. Rizvi
Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 6064, 606419, © 2006 SPIE-IS&T · 0277-786X/06/$15

priors for GACs. Kernel PCA was presented by Scholkopf[10] and allows to combine the precision of kernel methods with the reduction of dimension in the training set.

In the first section of this paper, we briefly recall generalities concerning active contours using level-sets. In the following section, we propose a consistent method to introduce shape priors within the GAC framework, using linear and Kernel PCA. In the third section, we compare performances obtained using both learning methods.

2. LEVEL-SET EVOLUTION

Level set representations were introduced by Osher and Sethian[15] in the field of computational physics and became a popular tool in the fields of image processing and computer vision. The idea consists of representing a contour by the zero level set of a smooth continuous function. A widespread choice is to use a signed distance function for embedding the contour. The contour is propagated implicitly by evolving the embedding function to decrease a chosen energy functional. Implicit representations present the advantage of avoiding to deal with complex re-sampling schemes of control points. Moreover, the contour represented implicitly can naturally undergo topological changes such as splitting and merging.

In what follows, the signed distance function used to represent the contour of interest will be denoted by Φ. The signed distance function Φ is a function $\Phi : \Omega \mapsto R$ and the contour corresponds to the zero level set of Φ, i.e. $\{(x, y) \in \Omega / \Phi(x, y) = 0\}$. All the information about the shape of the contour is embedded in the signed distance function Φ.

Starting with an arbitrary initial contour represented by an initial signed distance function $\Phi(0)$, we want to evolve the contour (i.e. knowing $\Phi(t)$, compute $\Phi = \Phi(t + dt)$) based on our knowledge of the shape of one or many objects of interest: This operation will be referred to as morphing in the rest of this paper. To this end, Φ will be updated in order to minimize an energy functional $E_{\text{shape}}(\Phi)$, embedding our knowledge about the shape of the object(s) of interest. E_{shape} will be chosen in order to be minimal when the curve adopts a shape that is consistent with our knowledge of shapes obtained a priori.

The minimization of E_{shape} with respect to Φ can be undertaken via usual gradient descent

$$\frac{d\phi}{dt} = -\nabla_\phi E_{\text{shape}} \quad i.e. \quad \Phi(t + dt) = \Phi(t) - dt.\nabla_\phi E_{\text{shape}}(\Phi(t)) \tag{1}$$

3. KERNEL PCA FOR SHAPE PRIOR

In this section, we first succinctly recall a general formulation allowing to perform linear PCA as well as Kernel PCA on any data set.[16, 17] Then we present specific kernels allowing to perform linear or non-linear principal component analysis on training sets of shapes. Finally, we propose an energy functional allowing to introduce shape priors obtained from either linear or Kernel PCA, within the GAC framework.

3.1. Kernel PCA

Kernel PCA can be considered to be a generalization of linear principal components analysis. This technique was introduced by Scholkopf,[10] and has proven to be a powerful method to extract nonlinear structures from a data set. The idea behind Kernel PCA consists of mapping a data set from an input space $\mathcal{I}$ into a feature space F via a possibly nonlinear function φ. Then, PCA is performed in F to find the orthogonal directions (principal components) corresponding to the largest variation in the mapped data set. Kernel PCA preserves the valuable learning properties obtained with principal component analysis. In particular, the first l principal components account for as much of the variance in the data as possible by using l directions. In addition, the error in representing any of the elements of the training set by its projection onto the first l principal components is minimal in the least square sense.

The nonlinear map φ typically does not need to be known, through the use of Mercer kernels. A *Mercer kernel* is a function $k(.,.)$ such that for all data points χ_i, the kernel matrix $\mathbf{K}(i, j) = k(\chi_i, \chi_j)$ is symmetric positive definite.[10] It can be shown that using $k(.,.)$ one can obtain the inner scalar product in F: $k(\chi_a, \chi_b) = (\varphi(\chi_a) \cdot \varphi(\chi_b))$, with $(\chi_a, \chi_b) \in \mathcal{I}$.

We now briefly describe the Kernel PCA method.[16] Let $\tau = \{\chi_1, \chi_2, ..., \chi_N\}$ be a set of training data. The centered kernel matrix $\tilde{\mathbf{K}}$ corresponding to τ, is defined as

$$\tilde{\mathbf{K}} = (\varphi(\chi_i) - \bar{\varphi} \cdot \varphi(\chi_j) - \bar{\varphi}) = (\tilde{\varphi}(\chi_i) \cdot \tilde{\varphi}(\chi_j)) = \tilde{k}(\chi_i, \chi_j), \ for\ i \in [|1, N|] \tag{2}$$

with $\bar{\varphi} = \frac{1}{N}\sum_{i=1}^{N}\varphi(\chi_i)$, $\tilde{\varphi}(\chi_i) = \varphi(\chi_i) - \bar{\varphi}$ being the centered map corresponding to χ_i and $\tilde{k}(.,.)$ denotes the centered kernel function. Since $\tilde{\mathbf{K}}$ is symmetric, using Singular Value Decomposition, it can be decomposed as

$$\tilde{\mathbf{K}} = \mathbf{U}\mathbf{S}\mathbf{U}^t \tag{3}$$

where $\mathbf{S} = \mathrm{diag}(\gamma_1, ..., \gamma_N)$ is a diagonal matrix containing the eigenvalues of $\tilde{\mathbf{K}}$. $\mathbf{U} = [\mathbf{u}_1, ..., \mathbf{u}_N]$ is an orthonormal matrix. The column-vectors $\mathbf{u}_i = [u_{i1}, ..., u_{iN}]^t$ are the eigenvectors corresponding to the eigenvalues γ_i's. Besides it can easily be shown that $\tilde{\mathbf{K}} = \mathbf{HKH}$, where $\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^t$. $\mathbf{1} = [1, ..., 1]^t$ is an $N \times 1$ vector.

Let $\mathbf{C}$ denote the covariance matrix of the elements of the training set mapped by $\tilde{\varphi}$. Within the Kernel PCA methodology, $\mathbf{C}$ does not need to be computed explicitly, only $\tilde{\mathbf{K}}$ needs to be known to extract features from the training set.[17] In what follows, we will refer to the subspace of the feature space F spanned by the first l eigenvectors of $\mathbf{C}$ as the PCA Space. The PCA Space is the subspace of F, obtained from learning the training data.

Let χ be *any* element of the input space $\mathcal{I}$. The projection of χ on the PCA Space will be denoted by $P^l\varphi(\chi)$. In this notation l refers to the first l eigenvectors of $\mathbf{C}$ used to build the PCA Space. In the feature space F, the squared distance d_F^2 between a test mapped point $\varphi(\chi)$ and its projection on the PCA Space is given by[10]:

$$d_F^2[\varphi(\chi), P^l\varphi(\chi)] = \parallel \varphi(\chi) - P^l\varphi(\chi) \parallel^2 = k(\chi,\chi) - 2\varphi(\chi)^t P^l\varphi(\chi) + P^l\varphi(\chi)^t P^l\varphi(\chi)$$

Using some matrices manipulations, this squared distance can be expressed only in terms of kernels as:

$$d_F^2[\varphi(\chi), P^l\varphi(\chi)] = k(\chi,\chi) + \frac{1}{N^2}\mathbf{1}^t\mathbf{K}\mathbf{1} - \frac{2}{N}\mathbf{1}^t\mathbf{k}_\chi + \tilde{\mathbf{k}}_\chi^t\mathbf{M}\tilde{\mathbf{K}}\mathbf{M}\tilde{\mathbf{k}}_\chi - 2\tilde{\mathbf{k}}_\chi^t\mathbf{M}\tilde{\mathbf{k}}_\chi \tag{4}$$

where, $\mathbf{k}_\chi = [k(\chi,\chi_1)\ k(\chi,\chi_2)\ , ..., k(\chi,\chi_N)]^t$, $\tilde{\mathbf{k}}_\chi = \mathbf{H}(\mathbf{k}_\chi - \frac{1}{N}\mathbf{K}\mathbf{1})$ and $\mathbf{M} = \sum_{i=1}^{N}\frac{1}{\gamma_i}\mathbf{u}_i\ \mathbf{u}_i^t$. Hence, the projection $P^l\varphi(\chi)$ needs not be computed explicitly in the expression of the distance presented in 4.

3.2. Kernel for linear PCA

In their seminal paper, Leventon *et al.*[2] presented a method to learn shape variations by performing PCA on a training set of shapes (represented by signed distance functions). Using the following kernel, in the formulation of PCA presented above and involving kernel functions, amounts to performing Linear PCA on SDFs:

$$k_{\mathrm{id}}(\Phi_i, \Phi_j) = (\Phi_i.\Phi_j) = \int\int \Phi_i(u,v)\Phi_j(u,v)du.dv \tag{5}$$

for all SDFs Φ_i and $\Phi_j : \mathbf{R}^2 \mapsto \mathbf{R}$. In the notation k_{id}, "id" stands for the identity function: when performing linear PCA the kernel used is the inner scalar product in input space, hence the corresponding mapping function is $\varphi = id$, and input space and feature space are identical.

3.3. Kernel for nonlinear PCA

Choosing a nonlinear kernel function $k(.,.)$ leads to performing nonlinear PCA (Kernel PCA). The exponential kernel has been a popular choice in the machine learning community and has proven to nicely extract nonlinear structures from data sets. Using SDFs for representing shapes, this kernel is given by

$$k_{\varphi_\sigma}(\Phi_i, \Phi_j) = e^{-\frac{\|\Phi_i - \Phi_j\|^2}{2\sigma^2}}, \tag{6}$$

where σ^2 is the variance parameter computed *a priori* and $\|\Phi_i - \Phi_j\|^2$ is the squared L_2-distance between two SDFs Φ_i and Φ_j. In the notation k_{φ_σ}, "φ_σ" refers to the nonlinear function mapping elements of the input space to the feature space; this nonlinear function, besides, depends on the choice of σ in the kernel.

This exponential kernel is one among many possible choice of Mercer kernels allowing to perform Kernel PCA: Other kernels, such as polynomial kernels for instance, could possibly be used to extract other specific features from the training set.[10]

3.4. Shape Prior for GAC

To include prior knowledge on shape in the GAC framework, we propose to use the projection on the PCA Space as a model and to minimize the following energy:

$$E_{\text{shape}}(\phi) := d_F^2[\varphi(\phi), P^l\varphi(\phi)] \tag{7}$$

In (7), φ refers to either id (if linear PCA is performed) or φ_σ (if Kernel PCA is performed). Minimizing E_{shape} amounts to updating the test shape Φ so that its image by φ is driven towards the PCA Space. By minimizing the distance between the mapped shape of the contour and the PCA Space, the contour itself is expected to evolve and adopt a shape that resembles the learnt shapes (this will be verified in the experiments presented below). Besides, a similar idea has been used for the purpose of pattern recognition.[10, 17]

Many authors proposed to minimize the distance between the current shape and the mean shape obtained from a training set. The assumption is indeed often made that the underlying distribution of familiar shapes is Gaussian.[2, 8, 9] Following this assumption, driving the curve toward the mean shape is a sensitive choice. Here, however, we purposefully chose to use the projection of the (mapped) current SDF to drive the evolution because we would like to deal with objects of different geometry in the training set (e.g. Figure 1 , third line: A training set of 4 words was used for experiments). When dealing with objects of very different shapes, the underlying distribution can be quite non-Gaussian (e.g.: multi-modal). Thus, the average shape would not be meaningful in this case, since it would amount to mixing shapes belonging to different clusters. As a consequence, driving the (mapped) current shape towards its projection onto the PCA Space appears to be a more sensible choice for our purpose.

The gradient of E_{shape} can be computed by applying calculus of variation on (4). For the kernel given in (5), corresponding to linear PCA, the following result is obtained:

$$\nabla_\phi E_{\text{shape}}^{\text{linear}} = 2\Phi + \Sigma_{i=1}^N g_i(\phi).\phi_i \tag{8}$$

with

$$[g_1(\phi), ..., g_N(\phi)] = -\frac{2}{N}\mathbf{1}^t + 2\tilde{\mathbf{k}}_\phi^t\mathbf{M}\tilde{\mathbf{K}}\mathbf{M}\mathbf{H} - 4\tilde{\mathbf{k}}_\phi^t\mathbf{M}\mathbf{H} \tag{9}$$

where $\tilde{\mathbf{k}}_\phi$, $\mathbf{M}$ and $\tilde{\mathbf{K}}$ are computed for the kernel k_{id}.

For the kernel given in (6), corresponding to nonlinear PCA, the following result is obtained:

$$\nabla_\phi E_{\text{shape}}^{\text{nonlinear}} = -\frac{\Sigma_{i=1}^N g_i(\phi)k_{\varphi_\sigma}(\phi, \phi_i)[\phi - \phi_i]}{\sigma^2} \tag{10}$$

with $[g_1(\phi), ..., g_N(\phi)]$ defined as in equation (9), but with $\tilde{\mathbf{k}}_\phi$, $\mathbf{M}$ and $\tilde{\mathbf{K}}$ computed for the exponential kernel.

The minimization of E_{shape} can then be undertaken as presented in equation (1) for any arbitrary contour, resulting in the morphing of the contour towards a familiar shape, as presented in next section.

4. EXPERIMENTS

The goal of this section is to compare the performances of Kernel PCA to linear PCA, as methods for introducing shape priors in the active contour framework using level sets. The fact that energy functional E_{shape} is consistently defined for both shape learning methods in Equation (7), ensures that performances obtained from applying linear PCA or Kernel PCA can be accurately compared: In what follows, we will morph the same initial shapes and compare the final contours obtained for both methods in terms of their resemblances to the elements of different training sets used.

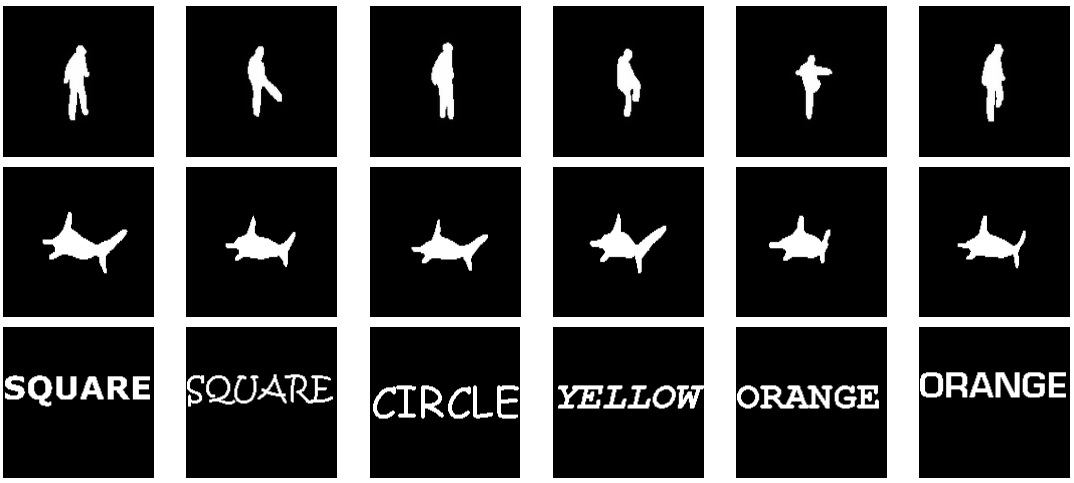


Figure 1. Three training sets (Before alignment - Binary images are presented here). First row, "Soccer Player" silhouettes (6 of the 22 used). Second row, "Shark" silhouettes (6 of the 15 used). Third row, "4 words" (6 of the 80 learnt; 20 fonts per word).

4.1. Morphing an arbitrary initial shape

4.1.1. Experimental protocol

For these experiments, one training set of shapes consisting of 22 shapes of a man playing soccer was used. The first line of Figure 1 presents a few elements of this Soccer Player training set: the binary maps corresponding to the training shapes are presented here, for clarity and comparison purposes. As a first step, these shapes were aligned using an appropriate registration scheme[3] to discard differences between them due to Euclidian transformations. Shape learning was then performed on both training sets as presented in Section 3: The familiar spaces of shapes were built for each of the kernels presented in equations (5) or (6), whether linear PCA or Kernel PCA was respectively performed. In both cases, the dimension of the PCA space was chosen to be 20 (C.f. Section 3.1, $l = 20$). The value $\sigma = 70$ was chosen in (6). Finally, Equation (1) was run until convergence, using the expression of the gradients presented in (8) for linear PCA and (10) for kernel PCA.

4.1.2. Comparative results

Figure 2 and Figure 3 present the morphing results obtained for two arbitrary initial shapes, one initial shape represents a house the other represents a "plus" sign. The first line of each of these figures shows the results obtained using kernel PCA. The second line of each of these figures presents the results obtained using linear PCA. As can be noticed, results obtained with linear PCA bear little resemblance with the elements of the training sets. By contrast, final contours obtained using kernel PCA are more faithful to the learnt shapes. Hence, Kernel PCA appears to outperform linear PCA as a method to introduce shape priors.

4.2. Robustness to misalignment

4.2.1. Experimental protocol

In these experiments, we intend to test the robustness of each framework (using linear and Kernel PCA) to misalignments relative to the registered shapes of a training set. This robustness to misalignment is an interesting feature to evaluate since, to perform segmentation, transformations (e.g.: translations) between the segmenting contour and the registered training shapes need to be taken into account. Hence, whether a probabilistic method[2] or a gradient descent scheme[3] is used to compensate for these transformations, more or less important misalignments between the contour and the registered training shapes may occur. This can impair the ability of the shape energy to properly constrain the shape of the segmenting contour.

The training set used in these experiments consists of 28 shapes representing a shark, as presented on Figure 1. These shapes were registered to compensate for Euclidian transformations. Shape learning and morphing were performed as presented above, using Kernel and linear PCA. The dimension of the PCA space was chosen to be 20 for both learning

methods. The value $\sigma = 70$ was chosen in (6). To test the robustness to misalignment, two elements of the training set were translated as compared to the registered training shapes and were used as initial contours for morphing. For the first initial contour used, a small translation of about 5 pixels in the x and y-directions was performed. The second initial contour was translated of about 15 pixels in the x and y-directions as compared to the registered training shapes.

4.2.2. Comparative results

Figure 4 and Figure 5 present the morphing results obtained for the slightly and more heavily misaligned initial shape, respectively. The first line of each of these figures shows the results obtained using Kernel PCA, the second line presents the results obtained using linear PCA. As can be noticed again, results obtained with linear PCA bear little resemblance with the elements of the training sets. By contrast, final contours obtained using Kernel PCA are more faithful to the learnt shapes. Besides, whether the initial contour is slightly or heavily misaligned, the contour evolution results in a translation of the initial contour, when Kernel PCA is used. Interestingly enough, the shape of the final contour obtained using Kernel PCA is very much alike the shape of the initial contour, in both situations. Hence, Kernel PCA appears to offer a higher level of robustness than linear PCA not only to slight but also to rather important misalignment of the initial contour.

4.3. Multi-modal learning

4.3.1. Experimental protocol

Kernel methods have been used to learn complex multi-modal distributions in an unsupervised fashion.[10] The goal of this section is to investigate the ability of Kernel PCA to simultaneously learn objects of different shapes and to constrain the contour evolution in a meaningful fashion. Besides, we want to contrast performances obtained with Kernel PCA to performances obtained with linear PCA. To this end, we built a training set consisting of four words, “orange”, “yellow”, “square” and “circle” each written using 20 different fonts. The size of the fonts was chosen to lead to words of roughly the same length. The obtained words (see Figure 1) were then registered according to their centroid. No further effort such as matching the letters of the different words was pursued. The linear and Kernel PCA methods presented in Section 3 were used to build the corresponding spaces of shapes for the registered binary maps. Diverse contours, which shapes bore some degree of resemblance to either of the 4 words (“orange”, “yellow”, “square” or “circle”), were then used as initial contours for morphing.

4.3.2. Comparative results

Each line in Figure 6 presents the morphing results obtained for the diverse initial contours used, for both linear and Kernel PCA.

The shape of the initial contour presented on the first line of Figure 6 is the word “square”, in which letters are partially erased. The morphing result using linear PCA bears little resemblance with the word “square”. By contrast, using Kernel PCA, the word square is accurately reconstructed. In particular a police close to the original one used in the initial contour is obtained.

The shape of the initial contour presented on the second line of Figure 6 is the word “circle”, occluded by a line. Again, the morphing result using linear PCA bears little resemblance with the word “circle”. In fact the obtained result appears as a mixing between different words. Using Kernel PCA, the word “circle” is not only accurately reconstructed but the line is completely removed. Besides, the original police used for the initial contour (which belongs to the training set) is preserved.

The shape of the initial contour presented on the third line of Figure 6 is one of the registered training words “yellow” and slightly translated. The letter “y” is also replaced by a rectangle. Using Kernel PCA, the word “yellow” is perfectly reconstructed: the letter “y” is recovered, the contour is translated back and the original police used for the initial contour is preserved. In comparison, the word “yellow” is barely recognizable from the final contour obtained using linear PCA.

In each of the experiment above, the accurate word (i.e.: closest to the word used to build the initial contour) is detected and reconstructed, using Kernel PCA. Hence, the method involving nonlinear Kernel PCA leads to better performnaces than the one involving linear PCA: The shape of the final contour obtained with linear PCA, is indeed oftentimes the result of a mixing between words of different classes. This mixing between classes can lead to unrealistic shapes. Thus, Kernel PCA appears as a superior method to introduce shape priors within the GAC framework, when training sets involving different types of shapes are used.

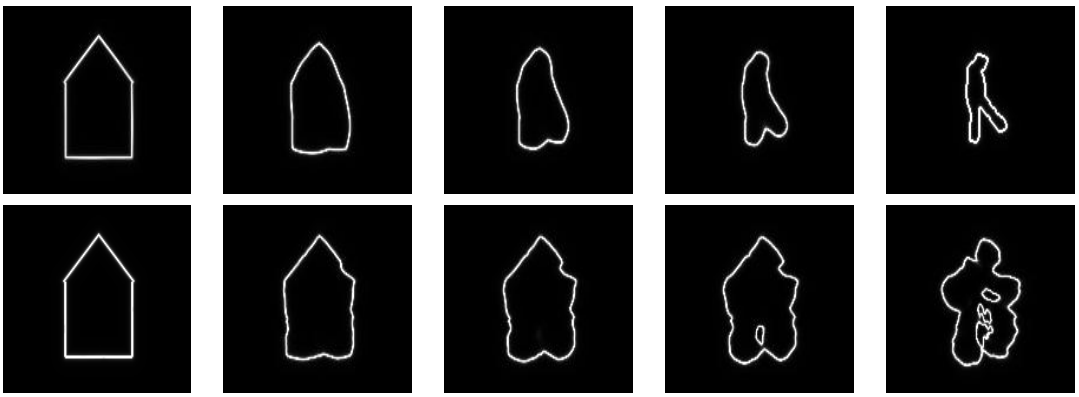


Figure 2. Morphing Results obtained for the "Soccer Player" training set, starting with an initial contour representing a house. First line: Evolution obtained using Kernel PCA; Second line: Evolution obtained using linear PCA. When kernel PCA is used, the final contour resembles the elements of the training set. By contrast, final contour obtained using linear PCA bears little resemblance with the learnt shapes.

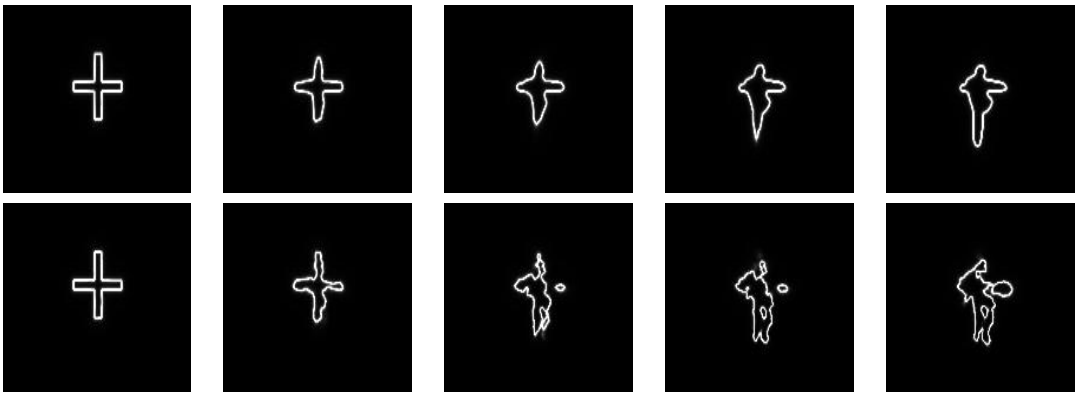


Figure 3. Morphing results obtained for the "Soccer Player" training set, starting with an initial contour representing a "plus" sign. First line: Evolution obtained using Kernel PCA; Second line: Evolution obtained using linear PCA. When kernel PCA is used, the final contour resembles the elements of the training set. By contrast, final contour obtained using linear PCA bears little resemblance with the learnt shapes.

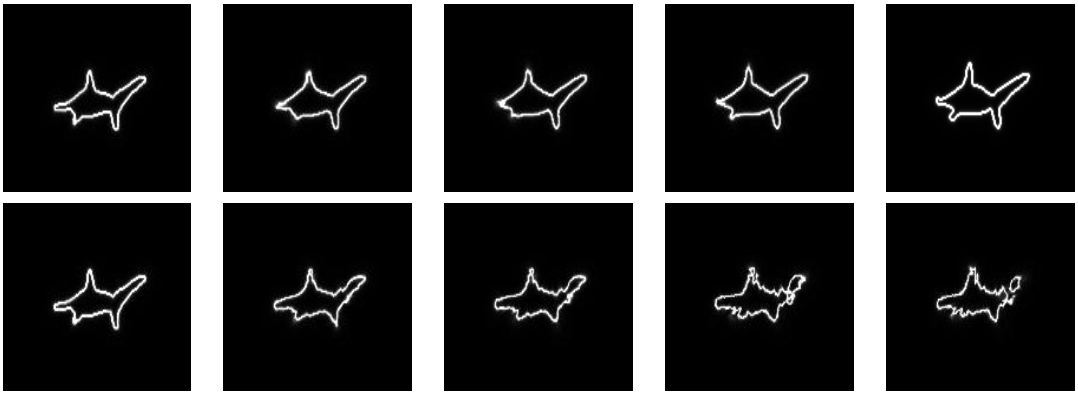
5. CONCLUSION

In this paper, we presented a framework to introduce shape priors for geometric active contours, which relies on a powerful and unsupervised statistical learning technique. The proposed framework uses Kernel principal component analysis and compares advantageously to a currently available method (using linear PCA) from the precision and computational-cost point of view. The proposed framework was shown to be robust to misalignments and to be able to deal efficiently with multi-modal distributions of shapes in the training sets.

In our future work, we plan to study the influence of other types of kernels (such as polynomial kernel, for instance) on the performances of the proposed framework. Besides, we plan to combine the proposed shape energy to energies encoding image information in order to perform segmentation. The superior performances obtained with Kernel PCA, as a means to constrain the contour to adopt familiar shapes, are expected to lead to robust segmentation performances.

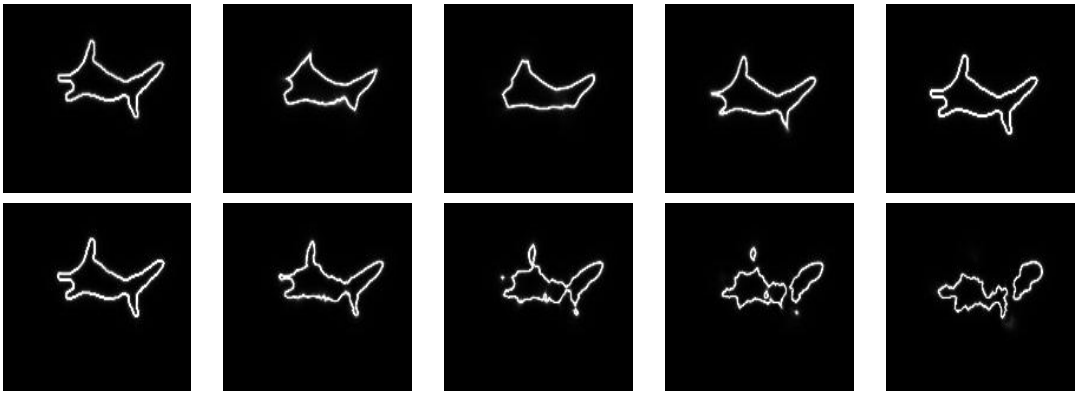
REFERENCES

1. N. Paragios and R. Deriche, "Geodesic active contours and level sets for the detection and tracking of moving objects," in *Transactions on Pattern analysis and Machine Intelligence*, **22**, pp. 266–280, 2000.



(a) (b) (c) (d) (e)

Figure 4. Morphing results obtained for the "Shark" training set. the initial contour (leftmost image) represents one of the learnt shapes slightly misaligned (4 pixels) with the corresponding element of the registered training set. First line: Evolution obtained using Kernel PCA; Second line: Evolution obtained using linear PCA. The final contour resembles obtained using Kernel PCA resembles the elements of the training set and is positioned where the learnt shapes were registered. The final contour obtained using linear PCA bears little resemblance with the learnt shapes.



(a) (b) (c) (d) (e)

Figure 5. Morphing results obtained for the "Shark" training set. The initial contour (leftmost image) represents one of the learnt shapes misaligned of approximately 15 pixels. First line: Evolution obtained using Kernel PCA; Second line: Evolution obtained using linear PCA. When kernel PCA is used, the final contour not only resembles the elements of the training set but is positioned where the learnt shapes were registered. By contrast, the final contour obtained using linear PCA bears little resemblance with the learnt shapes.

(a) (b) (c)

Figure 6. Morphing Results for the "4 Words" training set. (a): Initial contour; (b): Morphing using linear PCA; (c): Morphing using kernel PCA

2. M. Leventon, E. Grimson, and O. Faugeras, "Statistical shape influence in geodesic active contours," in *Proc. CVPR*, pp. 1316–1324, IEEE, 2000.
3. A. Tsai, T. Yezzi, and W. W. et al., "A shape-based approach to the segmentation of medical imagery using level sets," *IEEE Trans. on Medical Imaging* **22**(2), pp. 137–153, 2003.
4. A. Yezzi, S. Kichenassamy, and A. K. et al., "A geometric snake model for segmentation of medical imagery," in *IEEE Trans. Medical Imag.*, **16**, pp. 199–209, 1997.
5. A. Yezzi and S. Soatto, "Deformotion: Deforming motion, shape average and the joint registration and approximation of structures in images," in *International Journal of Computer Vision*, **53**, pp. 153–167, 2003.
6. T. Cootes, C. Taylor, and D. C. et al., "Active shape models-their training and application," in *Comput.Vis. Image Understanding*, **61**, pp. 38–59, 1995.
7. Y. Wang and L. Staib, "Boundary finding with correspondance using statistical shape models," in *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 338–345, 1998.
8. D. Cremers, T. Kohlberger, and C. Schnoerr, "Diffusion snakes: introducing statistical shape knowledge into the mumford-shah functional," in *International journal of computer vision*, **50**.
9. D. Cremers, T. Kohlberger, and C. Schnoerr, "Shape statistics in kernel space for variational image segmentation," in *Pattern Recognition*, **36**, pp. 1292–1943, 2003.
10. S. Mika, B. Scholkopf, and A. S. et al., "Kernel pca and de-noising in feature spaces," in *Advances in neural information processing systems*, **11**, 1996.
11. G. Sapiro, ed., *Geometric Partial Differential Equations and Image Analysis*, Cambridge University Press, 2001.
12. S. J. Osher and J. A. Sethian, "Fronts propagation with curvature dependent speed: Algorithms based on hamilton-jacobi formulations," *Journal of Computational Physics* **79**, pp. 12–49, 1988.
13. M. Rousson and N. Paragios, "Shape priors for level set representations," in *Proceedings of European Conference on Computer Vision*, pp. 78–92, 2002.
14. D. Cremers, S. Osher, and S. Soatto., "Kernel density estimation and intrinsic alignment for knowledge-driven segmentation: teaching level sets to walk," in *Pattern Recognition Proc. of DAGM 2004*, **3157**, pp. 36–44, 2004.
15. J. Sethian, *Level Set Methods and Fast Marching Methods*, 1999.
16. J. Kwok and I. Tsang, "The pre-image problem in kernel methods," in *IEEE transactions on neural networks*, **15**, pp. 1517 –1525.
17. B. Scholkopf, S. Mika, and A. Smola, "Nonlinear component analysis as a kernel eigenvalue problem," in *Neural Computation*, **10**, 1998.